漸進的な機械翻訳のための文法規則の変換手法

浅井 悟　　松原 茂樹　　外山 勝彦　　稲垣 康善

名古屋大学大学院工学研究科

{asai,matu,toyama,inagaki}@inagaki.nuie.nagoya-u.ac.jp

1 はじめに

効率的な話し言葉翻訳システムの実現のため，著者らは，漸進的な機械翻訳について研究を進めており，これまでに，文法的不適格表現を活用した漸進的な英日話し言葉翻訳手法を提案している [5, 6]．この手法は，解析処理および変換処理の2つの過程から構成されており，それらが英語音声の入力に対して同期的に処理を実行する．このうち解析処理では，即時的な変換処理を実現するために，文の入力途中の段階で随時，それまでの入力に対する構文構造を作成する．しかし，未入力部分が存在する段階で構文構造を決定するために，それ以降の入力をある程度予測する必要があり，一文単位の処理に比べて，生成される構文構造の数が増加し，解析処理時間が増大するという問題がある．

解析処理により生成される構文構造の数の増加は，同時に，変換処理対象となる選択肢の増加を意味する．漸進的な翻訳処理システムでは，即時的な翻訳を目的としているので，解析処理の途中で作成される複数の構文構造の中から敢えて1つを選択し，変換処理を実行することが不可欠であるが，選択肢が増加すると誤った構文構造を選択する可能性が高くなる．これに対して著者らは，たとえ選択に誤った場合でも，それが判明した時点で言い直しを生成することにより，適切な日本語を出力する方法を提案している [1]．しかし，言い直しの過度の出現は翻訳文全体の意味を理解しにくくする恐れがあり，言い直しはできる限り避けることが望ましい．

そこで本稿では，漸進的な翻訳処理において用いる文法規則を，等価で，かつ漸進的な解析処理で生成される構文構造の数が抑制され得るような文法規則に変換する手法を提案する．変換された文法規則を利用することにより，解析処理時間を短縮することができる．また，変換処理において誤った構文構造を選択する可能性が低下するため，言い直しは減少し，翻訳精度の向上が期待できる．

2 漸進的な翻訳処理とその問題点

2.1 漸進的なチャート解析法

著者らが開発した漸進的な英日話し言葉翻訳システム [6] の構成を図1に示す．本システムの漸進的な解析処理として，チャート [3] を用いた手法を採用している [4]．この手法では，i 番目の語 w_i の入力に対して，まず，辞書引きし，次に，それまでに入力された語の系列 $w_1w_2\cdots w_{i-1}$ の構文構造を表すチャートの弧 S_{i-1} の最左未決定項の範疇を目標として，文法規則を可能な限り適用し，弧を生成する．さらに，S_{i-1} の最左未決定項を，生成した弧の項で置き換えることにより，系列 $w_1w_2\cdots w_i$ に対する弧 S_i を生成する．

これらの操作により，任意の i に対して，語 w_i が入力された時点で，節点0と i の間に範疇がSである弧を少なくとも一つ作成することができる．

2.2 生成される構文構造の数の問題

漸進的な解析処理では，語が入力されるたびに，それまでの入力に対する構文構造をもつ弧を作成する．しかし，文の入力途中の段階では，一文全体の完全な構文構造を決定するために必要な情報は当然ながら不足しており，また，それらを正確に予測することは困難である．このため，一文単位の解析処理に比べ，多くの構文構造が生成されることになる．例えば，図2に示した文脈自由文法を用いて "the book shelf" で始まる文を解析する場合，通常の一文単位の解析処理では，"the" を処理する時点において，それ以降の入力語に関する構文情報を利用できるため，構文構造 (2) のみを作成することも可能である．これに対して，漸進的な解析処理では，それらの情報を利用できないため，"the" が入力された段階で以下の3つの構文構造が生成されることになる．

$$[[[\mathit{the}]_{\mathrm{Det}}\ [?]_{\mathrm{Adj}}\ [?]_{\mathrm{N}}]_{\mathrm{NP}}\ [?]_{\mathrm{VP}}]_{\mathrm{S}} \quad (1)$$

$$[[[\mathit{the}]_{\mathrm{Det}}\ [?]_{\mathrm{N}}\ [?]_{\mathrm{N}}]_{\mathrm{NP}}\ [?]_{\mathrm{VP}}]_{\mathrm{S}} \quad (2)$$

$$[[[\mathit{the}]_{\mathrm{Det}}\ [?]_{\mathrm{N}}\ [?]_{\mathrm{PP}}]_{\mathrm{NP}}\ [?]_{\mathrm{VP}}]_{\mathrm{S}} \quad (3)$$

構文構造の数の増加により，解析時間は増大する．したがって，これを抑制する手段が必要となる．

一方，変換処理では，解析処理で生成された構文構造の中から一つを選択し，それに対して変換規則を適用することにより，日本語翻訳結果を作成する．生成された構文構造はいずれも，変換処理対象の候補となる．これに対して著者らは，変換処理可能な構文構造が複数存在する場合でも，その中から敢えて1つを選択し，それを変換するという手法を採用している [5]．この手法では，たとえ構文構造の選択を誤ったとしても，それが判明した時点で，それまでに生成された日本語を言い直すことにより，正しく意味内容の通じる程度の翻訳結果を出力することができる [1]．しかし，変換処理可能な構文構造の数が増加すると，誤った構文構造を選択する可能性が高くなる．例えば，"the" の解析処理によって生成された構文構造 (1),(2),(3) に対して，(1) が選択され，変換処理が実行されたとする．このとき，次の入力語 "book" を解析すると，(1) については，最左未決定項 $[?]_{\mathrm{Adj}}$ と置き換え可能な項を生成することができないため，(2),(3) をもとにそれぞれ，(4),(5) を生成する．

$$[[[\mathit{the}]_{\mathrm{Det}}\ [\mathit{book}]_{\mathrm{N}}\ [?]_{\mathrm{N}}]_{\mathrm{NP}}\ [?]_{\mathrm{VP}}]_{\mathrm{S}} \quad (4)$$

$$[[[\mathit{the}]_{\mathrm{Det}}\ [\mathit{book}]_{\mathrm{N}}\ [?]_{\mathrm{PP}}]_{\mathrm{NP}}\ [?]_{\mathrm{VP}}]_{\mathrm{S}} \quad (5)$$

これは，構文構造 (1) の選択が誤りであったことを意味しており，この段階で (1) は棄却され，言い直しを生成する．さら

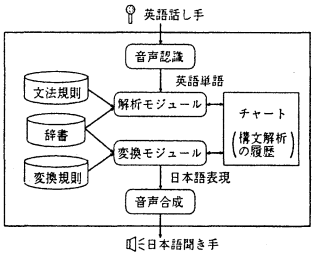

図 1: 漸進的な英日話し言葉翻訳システム

(g1) S → NP VP
(g2) NP → Det Adj N
(g3) NP → Det N N
(g4) NP → Det N PP
(g5) VP → V NP

図 2: 文法規則 (1)

(g1) S → NP VP
(g2) NP → Det X
(g3) X → Adj N
(g4) X → N N
(g5) X → N PP
(g6) VP → V NP

図 3: 文法規則 (2)

(g1) S → NP VP
(g2) NP → Det X
(g3) X → Adj N
(g4) X → N Y
(g5) Y → N
(g6) Y → PP
(g7) VP → V NP

図 4: 文法規則 (3)

(tg1) S → (NP 1 は NO) (VP 2 nil NO)
(tg2) NP → (Det 1 nil NO) (Adj 2 nil NO) (N 3 nil NO)
(tg3) NP → (Det 1 nil NO) (N 2 nil NO) (N 3 nil NO)
(tg4) NP → (Det 2 nil NO) (N 3 nil NO) (PP 1 nil NO)
(tg5) VP → (V 1 nil NO) (NP 2 を YES)

図 5: 変換規則

に，構文構造 (4),(5) から (5) が選択されれば，次の “shelf” の入力により，選択が誤りであったことがわかり，言い直しを生成することになる．

言い直しの頻出は，翻訳文全体の意味を分かりにくくする原因となる．よって，漸進的な翻訳処理の精度向上のためには，それ以降の入力語の情報を用いることなく，作成する構文構造の数を削減する方法が必要となる．

3 文法規則の変換手法

本節ではまず，解析処理で生成される構文構造の数を削減するために，文法規則をある特殊な形式に変換するというアイデアについて述べる．次に，文法を変換するための方法について述べ，そのアルゴリズムを提示する．なお，漸進的な英日翻訳システムでは文脈自由文法を用いていることから，以下でも文脈自由文法を対象として議論を進める．

3.1 基本的なアイデア

変換処理可能な構文構造の数を削減するための基本的なアイデアは以下の通りである．例えば，構文構造 (1),(2),(3) はそれぞれ，構造 $[the]_{\mathrm{Det}}$ に対して図 2の文法規則 (g2),(g3),(g4) を適用することにより得られる構文構造

$$[[the]_{\mathrm{Det}}\ [?]_{\mathrm{Adj}}\ [?]_{\mathrm{N}}]_{\mathrm{NP}} \quad (6)$$
$$[[the]_{\mathrm{Det}}\ [?]_{\mathrm{N}}\ [?]_{\mathrm{N}}]_{\mathrm{NP}} \quad (7)$$
$$[[[the]_{\mathrm{Det}}\ [?]_{\mathrm{N}}\ [?]_{\mathrm{PP}}]_{\mathrm{NP}} \quad (8)$$

を用いて生成される．すなわち，文法規則の適用の段階で生成される構造を削減することができれば，変換処理可能な構文構造の削減も可能となる．ここで，構文構造 (6), (7), (8) に着目すると，それらはいずれも範疇 Det の語 “the” 以降に入力される語の範疇を予測した形になっていることがわかる．例えば (6) は，次に範疇 “Adj(形容詞)” の語が入力されることを予測した構造となっている．

そこで，図 2に示した文法規則を，範疇 X,Y を新たに導入することによって図 3の文法規則に変換し，それを用いて解析処理を行うことを考える．このとき，構造 $[the]_{\mathrm{Det}}$ に対する文法規則の適用の結果，構文構造 $[[the]_{\mathrm{Det}}\ [?]_{\mathrm{X}}]_{\mathrm{NP}}$ が生成され，これを用いると変換処理可能な構文構造は

$$[[[the]_{\mathrm{Det}}\ [?]_{\mathrm{X}}]_{\mathrm{NP}}\ [?]_{\mathrm{VP}}]_{\mathrm{S}} \quad (9)$$

だけとなる．なお，図 2の文法と図 3の文法は等価である．

さて，次に入力された語 “book” に対する構造 $[book]_{\mathrm{N}}$ から，以下の 2 つの構文構造が生成される．

$$[[[the]_{\mathrm{Det}}\ [[book]_{\mathrm{N}}\ [?]_{\mathrm{N}}]_{\mathrm{X}}]_{\mathrm{NP}}\ [?]_{\mathrm{VP}}]_{\mathrm{S}} \quad (10)$$
$$[[[the]_{\mathrm{Det}}\ [[book]_{\mathrm{N}}\ [?]_{\mathrm{PP}}]_{\mathrm{X}}]_{\mathrm{NP}}\ [?]_{\mathrm{VP}}]_{\mathrm{S}} \quad (11)$$

これに対しても，図 3に示した文法規則 (g4),(g5) を図 4の文法規則 (g4),(g5),(g6) に変換し，その中の (g4) を用いて解析することにより，作成される変換処理可能な構文構造は次の一つのみとなる．

$$[[[the]_{\mathrm{Det}}\ [[book]_{\mathrm{N}}\ [?]_{\mathrm{Y}}]_{\mathrm{X}}]_{\mathrm{NP}}\ [?]_{\mathrm{VP}}]_{\mathrm{S}} \quad (12)$$

このように，文法規則を変換することにより，変換処理対象の候補数が削減され，対象の選択に誤る可能性も低下し，結果として，言い直しの生成頻度も減少する．

入力 “*the book shelf*” を図 2の文法と図 4の文法を用いた解析結果の比較を表 1に示す．変換された後の文法を用いることにより，入力語ごとに生成される構文構造の数が削減していることが分かる．

3.2 文法規則の変換手続き

文法規則の変換手続きを以下に示す．

変換手続き　左辺の範疇が等しくて，かつ，右辺の先頭から k 番目までの範疇の系列が等しい任意の文法規則

$$\begin{aligned} X &\rightarrow Y_1 Y_2 \ldots Y_k Z_{11} \ldots Z_{1n_1} \\ &\vdots \\ X &\rightarrow Y_1 Y_2 \ldots Y_k Z_{m1} \ldots Z_{mn_m} \end{aligned}$$

に対して，“$Z_{11} \cdots Z_{1n_1}$”,$\cdots$,“$Z_{m1} \cdots Z_{mn_m}$” に書換え可能とする範疇 W を新たに導入することにより，それを以下

表 1: 変換前の文法と変換後の文法によって生成される構文構造の数の比較

	変換前の文法	構文構造の数	変換後の文法	構文構造の数
the	$[[the]_{\mathrm{Det}}\ [?]_{\mathrm{Adj}}\ [?]_{\mathrm{N}}]_{\mathrm{NP}}$ $[[the]_{\mathrm{Det}}\ [?]_{\mathrm{N}}\ [?]_{\mathrm{N}}]_{\mathrm{NP}}$ $[[the]_{\mathrm{Det}}\ [?]_{\mathrm{N}}\ [?]_{\mathrm{PP}}]_{\mathrm{NP}}$	3	$[[the]_{\mathrm{Det}}\ [?]_{\mathrm{NEW}_1}]_{\mathrm{NP}}$	1
book	$[[the]_{\mathrm{Det}}\ [book]_{\mathrm{N}}\ [?]_{\mathrm{N}}]_{\mathrm{NP}}$ $[[the]_{\mathrm{Det}}\ [book]_{\mathrm{N}}\ [?]_{\mathrm{PP}}]_{\mathrm{NP}}$	2	$[[the]_{\mathrm{Det}}\ [[book]_{\mathrm{N}}[?]_{\mathrm{NEW}_2}]_{\mathrm{NEW}_1}]_{\mathrm{NP}}$	1
shelf	$[[the]_{\mathrm{Det}}\ [book]_{\mathrm{N}}\ [shelf]_{\mathrm{N}}]_{\mathrm{NP}}$	1	$[[the]_{\mathrm{Det}}\ [[book]_{\mathrm{N}}[[shelf]_{\mathrm{N}}]_{\mathrm{NEW}_2}]_{\mathrm{NEW}_1}]_{\mathrm{NP}}$	1

の文法規則

$$
\begin{aligned}
X &\rightarrow Y_1 Y_2 \ldots Y_k \\
W &\rightarrow Z_{11} \ldots Z_{1n_1} \\
&\vdots \\
W &\rightarrow Z_{m1} \ldots Z_{mn_m}
\end{aligned}
$$

で置き換える. ■

この手続きによって得られる文法は，変換前の文法と等価である．また，1回の手続きにより文法規則は1つ増加する．

3.3 文法規則と変換規則

これまでに開発した翻訳システム [6] では，図5のような変換規則を用いている．変換規則の右辺は，範疇，範疇の変換順序，範疇の変換処理後に補う助詞，および動詞の繰り返しの有無の4項組の列である．

文法規則と変換規則とが1対1に対応しているため，文法規則を変換すると，それに対応する変換規則も同様に変換する必要がある．文法規則は変換規則から容易に生成可能できることから，本稿ではまず，変換規則を変換し，それに対応する文法規則を作成することにより，文法規則の変換を実現する．

3.4 変換アルゴリズム

変換規則を変換するために，範疇だけでなくそれ以外の情報も考慮する必要がある．3.2節で示した変換手続きをおいて，各文法規則の範疇の系列 "$Y_1 Y_2 \ldots Y_k$" が等しくてもそれに対する変換処理の順序が異なれば，変換により順序に関する情報が破壊されることになる．したがって本稿では，変換順序が右回りに循環している変換規則のみを変換の対象とする．ここで，変換順序が右回りに循環している変換規則とは，次の条件を満たす変換規則である．

- 隣り合う2つの範疇の変換順序について，常に右側の範疇の変換順序の方が1つだけ大きい．ただし，左側の範疇の変換順序の方が大きい組が1組あってもよい．

例えば，図5の変換規則は，すべて変換順序が右回りに循環している変換規則である．変換順序が右回りに循環していない変換規則の例を図6に示す．

変換アルゴリズムを図7に示す．なお，ここでは簡単のため，範疇の変換処理後に補う助詞，動詞の繰り返しの有無の2つの項をまとめて e で表わす．まず最初に，変換順序が右回りに循環している変換規則のみを抽出する **(Step 0)**．規則の置き換えには，左辺の範疇が等しいなどの条件を満たす必要があるため，その条件に従って規則をいくつかの集合に分割する **(Step 1)**．また，ただ1つの規則からなる集合においては，変換を実行してもこれ以上生成される構文構造の数を減少させることはできないため，これを除外する **(Step 2)**．置き換えに関しては，各集合ごとに新しい範疇を2つ用意し，集合内の各規則に対して置き換えを実行する **(Step 3)**．以下これを繰り返す．

(tg6) NP → (N 1 nil NO) (PP 3 nil NO) (PP 2 nil NO)
(tg7) NP → (N 3 nil NO) (PP 2 nil NO) (PP 1 nil NO)
(tg8) NP → (Det 1 nil NO) (Adj 2 nil NO) (N 4 nil NO)

図 6: 変換順序が右回りに循環していない変換規則

4 評価

4.1 実験の概要

本手法の有効性を確認するために，アルゴリズムの適用前と適用後の変換規則を使ってそれぞれ翻訳実験を行い，生成される弧の数，解析処理時間，言い直しの頻度，及び翻訳精度を調べた．翻訳システムは，GNU Common Lisp 2.2.2 を用いて UltraSPARC-II (メインメモリ 512MB，CPU クロック 248MHz) 上に実装している [6]．コーパスとして ATR 対話データベース [2] を使用した．対話領域は旅行の申込みであり，英語話者による発話 278 文を入力とした．入力文の平均文長は 7.1 語である．変換規則および辞書を本実験のために新たに作成した．辞書の規模は 544 語，変換前の変換規則数は 185 規則であり，変換後の変換規則数は 221 規則となった．

4.2 実験結果

実験結果を表2に示す．文法規則の変換により，一単語が入力されたときの弧の平均作成個数は，541.7 個から 57.9 個に減少した．一単語に対する平均解析処理時間は，変換アルゴリズムの適用前では 2.946 秒であったのに対し，変換アルゴリズムの適用後では 0.046 秒であった．また，一文に対する平均言い直し生成回数は，約1回減少した．このことから，本手法の有効性を確認できた．

Grammar_Transfer(P)

begin

Step 0 変換順序が右回りに循環している変換規則の抽出:

for each *rule* ∈ P

if *rule* の変換順序が右回りに循環している **then**

P* := P* ∪ *rule*

else

P' := P' ∪ *rule*;

Step 1 P* の分割:

P* の *rule* に対して,

1) 左辺の範疇

2) 左隅の要素の範疇, 補う助詞, 動詞の繰り返しの有無

3) 左隅の要素の変換順序が 1 であるか否か

が同じものを $subset_i$ に分割する

Step 2 置き換え不可能な集合の削除:

for each $subset_i$ ∈ P*

if $|\ subset_i| = 1$ **then**

$subset_i$ 中の rule を P' に入れる

Step 3 置き換え:

for each $subset_i$

新しい範疇 NEW_1, NEW_2 を用意する

for each

$\alpha \rightarrow (\alpha_1\ o_1\ e_1)\cdots(\alpha_k\ o_k\ e_k)$
$(\alpha_{k+1}\ o_{k+1}\ e_{k+1})\cdots(\alpha_{j-1}\ o_{j-1}\ e_{j-1})$
$(\alpha_j\ o_j\ e_j)\cdots(\alpha_h\ o_h\ e_h)$
$(\alpha_{h+1}\ o_{h+1}\ e_{h+1})\cdots(\alpha_n\ o_n\ e_n) \in subset_i$

replace

$|\ \alpha \rightarrow (\alpha_1\ o_1\ e_1)\cdots(\alpha_k\ o_k\ e_k)$
$(\alpha_{k+1}\ o_{k+1}\ e_{k+1})\cdots(\alpha_{j-1}\ o_{j-1}\ e_{j-1})$
$(\alpha_j\ o_j\ e_j)\cdots(\alpha_h\ o_h\ e_h)$
$(\alpha_{h+1}\ o_{h+1}\ e_{h+1})\cdots(\alpha_n\ o_n\ e_n)|$

with

$|\ \alpha \rightarrow (\alpha_1\ o_h{+}2\ e_1)\cdots(\alpha_k\ o_h{+}k{+}1\ e_k)$
$(\alpha_{k+1}\ o_h{+}k{+}2\ e_{k+1})\cdots(\alpha_{j-1}\ o_h{+}j\ e_{j-1})$
$(\alpha_j\ o_j\ e_j)\cdots(\alpha_h\ o_h\ e_h)$
$(NEW_1\ o_{h+1}\ e_{NEW_1})$

$NEW_1 \rightarrow (\alpha_{h+1}\ o_{h+1}\ e_{h+1})\cdots(\alpha_n\ o_n\ e_n)|$;

replace

$|\alpha \rightarrow (\alpha_1\ o_h{+}2\ e_1)\cdots(\alpha_k\ o_h{+}k{+}1\ e_k)$
$(\alpha_{k+1}\ o_h{+}k{+}2\ e_{k+1})\cdots(\alpha_{j-1}\ o_h{+}j\ e_{j-1})$
$(\alpha_j\ o_j\ e_j)\cdots(\alpha_h\ o_h\ e_h)$
$(NEW_1\ o_{h+1}\ e_{NEW_1})|$

with

$|\ \alpha \rightarrow (\alpha_1\ o_h{+}2\ e_1)\cdots(\alpha_k\ o_h{+}k{+}1\ e_k)$
$(NEW_2\ o_h{+}k{+}2\ e_{NEW_2})\cdots(\alpha_{j-1}\ o_h{+}j\ e_{j-1})$
$(\alpha_j\ o_j\ e_j)\cdots(\alpha_h\ o_h\ e_h)$
$(NEW_1\ o_{h+1}\ e_{NEW_1})$

$NEW_2 \rightarrow (\alpha_{k+1}\ 1\ e_{k+1})\cdots(\alpha_{j-1}\ j{-}k\ e_{j-1})|$;

Step 4 終了条件のチェック:

if P* ≠ φ **then**

Step 1 へジャンプ

end.

変数の説明

P: 変換前の変換規則の集合
P*: 変換途中の変換規則の集合
P': 変換後の変換規則の集合
i: 汎用変数
j: 規則中の j 番目の要素の変換順序が 1
k:1 から k 番目の要素の範疇, 補う助詞, 動詞の繰り返しの有無が一致
h:j から h 番目の要素の範疇, 補う助詞, 動詞の繰り返しの有無が一致

図 7: 変換アルゴリズム

表 2: 実験結果

評価項目	変換前	変換後
弧の平均作成数 (個)	541.7	57.9
一単語に対する解析処理時間 (秒)	2.946	0.046
一文の言い直し回数 (回)	3.21	2.11
翻訳成功率 *(%)	72.7	65.1

また, 翻訳精度について調べた. 正しく意味の通じる程度の品質を備えた翻訳結果が得られた原言語文を翻訳成功とし, その入力文数に対する割合を翻訳成功率とした. 変換前に比べ, 翻訳成功率は 7.6% 上昇した. これは言い直しの減少により, 翻訳結果の品質が向上したことを意味している.

5 おわりに

本稿では, 漸進的な解析処理において生成される構文構造の数を抑制するための文法規則の変換手法について述べた. 翻訳実験により, 本手法の有効性を確認した. 本稿では, 変換順序が右回りに循環している変換規則に対応した文法規則のみを変換の対象とした. 任意の文法規則を漸進的な処理に適した文法規則に変換できるように本手法を拡張することは今後の課題である.

本手法は, より精度の高い漸進的な翻訳処理の実現を目的として開発されたが, アルゴリズムの若干の修正により, 自然言語の漸進的解釈において実行される構文解析においても有用であると考えられる. 解釈の対象とする構文構造が削減されるため, より効率的な意味解釈が実現が期待できる.

参考文献

[1] 浅井 悟, 松原 茂樹, 外山 勝彦, 稲垣 康善: 漸進的な英日話し言葉翻訳システムにおけるチャートに基づく変換手法, 情報処理学会研究報告 97-NL-122, pp. 29-34 (1997).

[2] 江原, 井ノ上, 幸山, 長谷川, 庄山, 森本: ATR 対話データベースの内容, テクニカルレポート TR-I-0186, ATR 自動翻訳電話研究所 (1990).

[3] Kay, M.: Algorithm Schemata and Data Structures in Syntactic Processing, *Technical Report CSL-80-12*, Xerox PARC (1980).

[4] 松原 茂樹, 浅井 悟, 外山 勝彦, 稲垣 康善: 漸進的な話し言葉翻訳のためのチャート解析法, 電気関係学会東海支部連合大会講演論文集, p. 555 (1997).

[5] 松原 茂樹, 浅井 悟, 外山 勝彦, 稲垣 康善: 不適格表現を活用する漸進的な話し言葉翻訳手法, 電気学会論文誌 C, Vol. 118-C, No. 1 pp. 71-78 (1998).

[6] Matsubara, S., Asai S., Inagaki, Y. and Toyama K.: Incremental Spoken Language Translation Utilizing Grammatically Ill-formed Expressions, *Proc. of 3rd Conf. of Pacific Association for Computational Linguistics*, pp. 188-194 (1997).